

Introduction

Belief revision is the process in which a cognitive model compares and adjusts its knowledge base to new incoming information. In the subject of belief revision, there is no finer application of this process than a model that can predict guiltiness of a suspect based on testimonies. This paper will outline the process that we went through to see this model into reality.

Background

The first step of realizing this model is research. We have found that not a lot of research has been done on our particular case, that is, a Guilt Machine where everyone is potentially lying and the killer is among them, and the Machine has to determine from these statements who is the killer.

We initially attempted to research strategies for the games Clue and Mafia, since we assumed our Guilt Machine would be similar to either a townspeople in a Mafia game or a random player in a Clue game. However, we've since realized that we're missing one important aspect from both these games that prevents these strategies from applying: multiple rounds. Unlike in the social deduction games, our program only has one 'round' of information gathering from everyone else. This means that we are unable to do any of the cross-round elimination that's integral to those strategies.

We then did research into how humans detect if someone is lying. Much of it is based on body language—for example, according to an article from Time Magazine, people who are lying tend to blink and fidget less and hesitate longer before speaking—but some of it is based on personal characteristics. That same article mentioned how the more intelligent and/or creative a person is, the more likely they are to lie. Conversely, the shorter and less detailed a story is, the more likely it is to be false (Barker). This means that we could theoretically have our Guilt Machine take into account the amount of terms in the testimony as an indicator of believability.

The reason for all of these symptoms of lying is because lying takes a larger cognitive load on someone than telling the truth. Therefore, many strategies involved in determining if someone is lying involve increasing that mental load, for example by asking them to repeat their story in reverse or by asking them for more details (Barker).

One article entitled *Distributed Belief Revision as Applied Within a Descriptive Model of Jury Deliberations* established many requirements and solutions for a project similar, albeit much more complex, than ours. The article specifically lays out a general approach to belief revision, which begins with its first principle; consistency. This means that revision must yield a consistent knowledge space. The next principle states that when revising, the change to the knowledge space must alter as little as possible, this means that we cannot remove any more facts from our knowledge base than what is the absolute minimum. Finally, incoming information must always belong to the revised knowledge space (Dragoni).

The article also listed requirements for a belief revision framework.. Some of the requirements this article lists are as follows; the framework must have the ability to reject incoming information and it must be able to recover previously discarded beliefs. The article goes on to explain how a system must be able to deal with coupled information rather than just the information alone, which means the algorithm must consider the source of the information alongside the information itself. Finally, the article describes how a system must be able to

combine contradictory and concomitant (things that go together) evidence (Dragoni). In the context of our project, this would just mean that new information must affect the old, as well as the old information must affect the new.

Another relevant detail that we had to take into account were the circumstances that may cause somebody to commit murder, which would allow us to build relationships and determine motives for suspects committing the crime, as well as motives to frame other suspects. Motives are typically a key aspect of murder investigations. Most reasonable people don't commit murder for no reason. Motives can often be really complex but one article lays it out really well. In general murders can be broken down into 4 "L"s, love, lust, loathing, and loot (Morrall). We broke down our possible motives into these categories. Our prolog facts of "married" and "have relationship" apply to the love and lust category at different levels and possibly loathing as well if it includes an affair. Our "owes_money" fact can represent loot and our "threatened" fact can represent loathing.

Our strategy pertaining to motives is to see which relationships would make somebody more likely to be the killer than somebody else, and which relationships would possibly cause somebody to fake a testimony to frame another specific suspect. In the end the motive can not determine guilt, but it can definitely influence the credibility of a suspect and the skepticism that we should have towards their testimony.

We also took into consideration that perhaps the suspect wasn't reasonable. In addition to the 4 "L's someone might kill somebody because they are a psychopath of some sort (Morrall). In our case we created the fact "arsonist" to represent this.

Methods

In order to successfully implement belief revision, we will need to outline our entire process, starting with how we will represent beliefs. First, we made the decision to represent all personal background knowledge as facts. This is to limit the scope of the project and reduce the overall complexity of the program. We will further limit the scope by only allowing a set number of weapons, motives, and suspects, and what they witnessed or heard. We will input testimonies one at a time as a tuple with elements representing what the witness saw, heard, what weapon they believe was used, who they believe the killer was, the motive, the name of the witness. An associated truth value will then be added. The only required value for the testimony will be who the witness believes is the killer.

Next, we must describe the internal reasoning that will be associated with the beliefs. We have decided to use a system of weights to associate the truth value with the testimony, and in turn, the confidence we have in who the killer is. Truth values will range from 0.1 to 1.1, with 0 being an internal contradiction, 0.1 being absolutely false, and 1.1 being absolutely true. The higher the value, the more confident the system will be. We can compare the testimonies to the information we receive from the bailiff, which we assume to be absolute truth, in order to derive our truth values. Assuming there is no internal contradiction, we can determine the truth value by the intersection of the testimony with our beliefs divided by the union of our beliefs plus our offset of 0.1. This simple system will allow us to get an accurate confidence value and begin to refine our knowledge base. We also will implement logic where if information contained in a testimony is accurate, it will both increase the truth value and make it more likely the witness

committed the crime for having knowledge about the crime scene. We will also have to incorporate motives into the credibility value.

Now we must indicate the conditions for revision. First and most obvious will be revising our knowledge based on contradictions between testimonies. If more witness' are saying the same thing, we can gradually replace old knowledge with more frequent testimonies.

Discussion

We developed 3 scenarios to test how well our computational model worked. Each scenario includes five separate testimonies that allow our model to revise its beliefs and deduce who the killer is. For all scenarios, our model successfully predicted who the killer was. The first scenario implied the killer was John and the bailiff's evidence said that the cause of death was poison and the motive was protection. However, we do not know this yet because the bailiff does not present their evidence until the end of all five testimonies. The first testimony was from John and he claimed that Craig killed the victim with poison because of adultery. All of John's claims appeared to make sense so after John's statement we believed that Craig was the killer and he killed the victim with poison. Next Craig testified and claimed that John was the killer with the exact same weapon of poison. Our beliefs did not change after this testimony, but it shifted the weight to make John slightly more likely to be the killer than he was before. The next testimony was given by Matthew. Matthew also claimed that John was the killer but he claimed that the weapon was rope rather than poison and the motive was protection. After this testimony our belief changes to thinking that John was the killer and he killed the victim with poison for protection. As you can see, our belief about the motive and killer changed but more evidence from the testimonies still suggested poison was the weapon so that did not change. The next testimony given was by Daniel, and none of the claims in Daniel's testimony matched any of the other testimonies at all. Because of this, our belief did not change at all from this testimony, but the weight for truth value was slightly shifted. The last testimony for this scenario was given by Theresa where she confirms the cause of death but states a different killer and motive. This does not change our belief about the killer or motive at all because there is still more support for the previous testimonies. In the end, our hypothesized killer for scenario one was correct. Similar results were recorded for the other scenarios and many different variations of them to test how the system reacted to certain details or uncertainties.

Our model seems to excel at matching testimonies and removing contradictions. Loads of time was spent to make sure that contradictions within testimonies would render the entire testimony useless. Also, as we discussed earlier, the model is phenomenal at deducing the guilty party.

While the model was an overall success, there are a few drawbacks that prevent it from being perfect in its decision making. The first of these drawbacks is that the order matters when taking the testimony. This means that the truth values associated with a testimony do not really matter until all of the testimonies are in. For example, since the model has no other previous knowledge, when the first testimony is entered, it is assumed to be mostly true as there is nothing to compare it with. This could easily change if the first testimony differs from the other four. The truth value or confidence level will decrease with each new and different testimony. Another drawback is how giving a correct testimony that agrees with the knowledge base

actually increases the likelihood of that particular witness being the killer. We believe that the current values we have are a bit too strong and this should not matter as much as it does.

A significantly more advanced version of this model could potentially be used in order to help deduce the likely perpetrator of a crime. However, this has been explored before with unsatisfactory results. A model such as this will always have some form of bias influencing its results making it not fair or just to use as a method of determining guilt. A form of this model could potentially be used to help narrow down the decision making process, but beyond that, it may not have much practical use. This is supported by the idea that as the scope is broadened, it only increases the chance that more unintentional biases will sneak into the model.

If we were to continue working with this model or rebuild it from scratch, we would expand the scope of the project to allow for more dynamic deductions. We are very limited in that a testimony can only give a handful of specific causes of death, weapons, or killers. Making a broader knowledge base of facts such as these would be interesting to implement into the program. A future addition to the model that would be fun to implement would be some form of natural language processing. This would make things more interesting in that a testimony could be given in a grammatically correct english sentence or statement. We could then parse out the whole testimony and revise our beliefs accordingly. Another thing we would have liked to have to implement would be for the model to take interpersonal relationships between suspects into account when deducing guilt. Right now, the model only looks at the relationships between the witnesses and the victim. This is potentially missing a huge piece of the puzzle as the relationships between the witnesses could provide motive, something our model does take into consideration.

Conclusion

Our solution did exactly as we planned; successfully modeled collections of information about a crime scene, and then revised those collections to accurately determine the guilty party. We determined this proof of concept could be taken farther with an expanded scope, but would likely yield little practical use due to the unintentional biases that develop in the implementation of a system such as this.

Bibliography

Lying:

Barker, Eric. "Signs Of Lying: Here's What Will And Will Not Help You Detect Lies." *Time*,

<https://time.com/77940/detect-lying/>. Accessed 2 Nov. 2021.

Heidenreich, Toni. "The Formal-Logical Characterisation of Lies, Deception, and Associated

Notions." *ResearchGate*, King's College London, Apr. 2013,

https://www.researchgate.net/publication/311969457_The_formal-logical_characterisation_of_lies_deception_and_associated_notions.

Florentine, Erica. "11 Ways To Tell If Someone Is Telling You The Truth, According To

Science." *Bustle*, 9 June 2016,

<https://www.bustle.com/articles/165346-11-ways-to-tell-if-someone-is-telling-you-the-truth-according-to-science>.

Murder Motives:

Morrall, Peter. "Murder and Society: Why Commit Murder?" *Centre for Crime and Justice*

Studies, Centre for Crime and Justice Studies, Winter 2006,

<https://www.crimeandjustice.org.uk/sites/crimeandjustice.org.uk/files/09627250608553401.pdf>.

Belief revision:

Dragoni, Aldo, et al. *Belief Revision as Applied Within a Descriptive Model of Jury*

Deliberations. ResearchGate, May 2000,

https://www.researchgate.net/publication/2599219_Distributed_Belief_Revision_as_Applied_Within_a_Descriptive_Model_of_Jury_Deliberations.

Appendix A: Code

%Visual:

%Blue face :- [rope, poison]

%Flames :- [matches]

%Convulsing :- [poison]

%Blood :- [gun, knife]

%Nothing :- [gun, knife, rope, poison, none]

%Audial:

%Gunshot :- [gun]

%Screaming :- [gun, knife, matches]

%Coughing :- [poison, matches]

%Banging :- [gun, knife, rope]

%Nothing :- [knife, rope, poison, matches, none]

% Weapons: [gun, knife, rope, poison, matches, none]

% Suspects: [john, craig, theresa, matthew, daniel, victim]

% All Weapons and their correlated symptoms

```

weapon(gun, Saw, Heard) :-      member(Saw, [blood, nothing]),
member(Heard, [gunshot, screaming, banging]).
weapon(knife, Saw, Heard) :-    member(Saw, [blood, nothing]),
member(Heard, [screaming, banging, nothing]).
weapon(rope, Saw, Heard) :-    member(Saw, [blue_face, nothing]),
member(Heard, [banging, nothing]).
weapon(poison, Saw, Heard) :-   member(Saw, [blue_face, convulsing, nothing]),
member(Heard, [coughing, nothing]).
weapon(matches, Saw, Heard) :-  member(Saw, [flames]),
                                member(Heard, [screaming, coughing, nothing]).
weapon(none, Saw, Heard) :-     member(Saw, [nothing]),
member(Heard, [nothing]).

```

```

% Relationship Info
married(craig, victim).
married(matthew, daniel).
have_relationship(theresa, victim).
have_relationship(craig, theresa).
have_relationship(john, victim).
have_relationship(daniel, victim).

```

```

% S has an affair on R with P if S is married to R, and S is in a relationship with P
affair(S, R, P) :- have_relationship(S, P), married(S, R), \+(R = P).

```

```

% owes_money(S, P) = S owes P money
% S = P implies S is in debt/owes money to the bank
owes_money(matthew, victim).
owes_money(victim, john).
owes_money(victim, victim).
owes_money(daniel, victim).
owes_money(john, john).

```

```

% arsonist(S) = S is an arsonist
arsonist(daniel).
arsonist(craig).
arsonist(victim).

```

```

% threatened(S, P) = S has threatened P
threatened(theresa, victim).
threatened(victim, craig).
threatened(craig, victim).
threatened(daniel, john).
threatened(victim, john).

```

```

motive(adultery, Suspect) :- affair(Suspect, victim, _); affair(victim, Suspect,
_).%married(Suspect, victim), (have_relationship(Suspect, !victim) | had_relationship(!Suspect,
victim)
motive(money, Suspect) :- owes_money(Suspect, victim); owes_money(victim, Suspect).
motive(pyromania, Suspect) :- arsonist(Suspect). %added this so insanity is really a catch-all
motive(protection, Suspect) :- threatened(Suspect, victim); threatened(victim, Suspect).
motive(suicide, victim) :- owes_money(victim, victim); affair(_, victim, _). % figure this out later
motive(insanity, _).

```

```

read_word_list(Ws) :-
    read_line_to_codes(user_input, Cs),
    atom_codes(A, Cs),
    tokenize_atom(A, Ws).

```

```

% Grammar for input
%Saw, Heard, Weapon, Killer, Motive, Owner
%the odd names is to avoid possible overlap with rules in the future
sentence(s(Testimony)) --> open_parens, testi(Testimony), close_parens.
sentence(s(Bailif)) --> open_parens, bail(Bailif), close_parens.
sentence(s(Testimony)) --> open_parens, testi(Testimony), close_parens, dot.
sentence(s(Bailif)) --> open_parens, bail(Bailif), close_parens, dot.

```

```

testi(t(Saw, Heard, Weapon, Killer, Motive, Witness)) --> eyes(Saw), comma, ears(Heard),
comma, weap(Weapon),
    comma, kill(Killer), comma, moti(Motive), comma, wit(Witness).

```

```

bail(b(Weapon, Motive)) --> weap(Weapon), comma, moti(Motive).

```

```

eyes(v(blue_face)) --> [blue,face]. %this is how prolog will match to 'blue face' (w/o the ' marks)
eyes(v(flames)) --> [flames].
eyes(v(convulsing)) --> [convulsing].
eyes(v(blood)) --> [blood].
eyes(v(nothing)) --> [nothing].
eyes(v(nothing)) --> [n, (/), a]. %this looks odd but this is how prolog will match to 'n/a' (w/o the '
marks)

```

```

ears(a(gunshot)) --> [gunshot].
ears(a(screaming)) --> [screaming].
ears(a(coughing)) --> [coughing].
ears(a(banging)) --> [banging].
ears(a(nothing)) --> [nothing].
ears(a(nothing)) --> [n, (/), a].

```

```
weap(w(gun)) --> [gun].
weap(w(knife)) --> [knife].
weap(w(ropes)) --> [ropes].
weap(w(poison)) --> [poison].
weap(w(matches)) --> [matches].
weap(w(none)) --> [none].
weap(w(none)) --> [n, (/), a].
```

```
kill(k(john)) --> [john].
kill(k(craig)) --> [craig].
kill(k(theresa)) --> [theresa].
kill(k(matthew)) --> [matthew].
kill(k(daniel)) --> [daniel].
kill(k(victim)) --> [victim].
```

```
moti(m(adultery)) --> [adultery].
moti(m(money)) --> [money].
moti(m(protection)) --> [protection].
moti(m(pyromania)) --> [pyromania].
moti(m(suicide)) --> [suicide].
moti(m(insanity)) --> [insanity].
moti(m(insanity)) --> [n, (/), a].
```

```
wit(o(john)) --> [john].
wit(o(craig)) --> [craig].
wit(o(theresa)) --> [theresa].
wit(o(matthew)) --> [matthew].
wit(o(daniel)) --> [daniel].
```

```
open_parens --> ['('].
close_parens --> [')'].
dot --> ['.'].
comma --> [', '].
```

```
output([Killer, Weapon, Motive]) :- write("I currently believe that "), write(Killer),
    write(" killed the victim with "), write_weapon(Weapon), write(" and that the motive was
"),
    write(Motive), write("."), nl.
```

```
%I went over the top formatting this for fun
write_weapon(gun) :- write(a), write(" "), write(gun).
write_weapon(knife) :- write(a), write(" "), write(knife).
write_weapon(ropes) :- write(ropes).
write_weapon(matches) :- write(matches).
```



```

write_weapon(poison) :- write(poison).
write_weapon(none) :- write("... actually, I "), write(don), write(""), write(t),
    write(" "), write(know), write(" what the murder weapon is").

start :- take_input([], [], 0).

take_input(Beliefs, Testimonies, 5) :- final_remarks(Beliefs, Testimonies), !.%final_remarks(),
output, !. (basically)
take_input(Beliefs, Testimonies, Count) :-
    Count < 5,
    read_word_list(Input),
    sentence(Parse, Input, []),
    process_testimony(Beliefs, Testimonies, Parse, NewTestimonies),
    revise_beliefs(Beliefs, NewTestimonies, NewBeliefs),
    recalculate_truth_values(NewTestimonies, NewBeliefs, FinalTestimonies),
    output(NewBeliefs),
    Count1 is Count + 1,
    take_input(NewBeliefs, FinalTestimonies, Count1).

% Rough idea of how input processing can go
process_testimony(Beliefs, Testimonies, Parse, NewTestimonies) :-
    extract(Parse, Statement),
    Statement = (Saw, Heard, Weapon, Killer, Motive, Witness),
    ((      internal_contradiction(Statement, Testimonies),
        append(Testimonies, [(0, Saw, Heard, Weapon, Killer, Motive, Witness)],
NewTestimonies));
    (      get_truth_value(Statement, Beliefs, TruthValue),
        append(Testimonies, [(TruthValue, Saw, Heard, Weapon, Killer, Motive, Witness)],
NewTestimonies))).

extract(s(t(v(Saw), a(Heard), w(Weapon), k(Killer), m(Motive), o(Witness))),
    (Saw, Heard, Weapon, Killer, Motive, Witness)).

extract(s(b(w(Weapon), m(Motive))), Weapon, Motive).

final_remarks([Killer, _, _], Testimonies) :-
    read_word_list(Input),
    sentence(Parse, Input, []),
    extract(Parse, Weapon, Motive),
    TempBeliefs = [Killer, Weapon, Motive],
    recalculate_truth_values(Testimonies, TempBeliefs, FinalTestimonies),
    %this is being weird and not grabbing everything, even though it grabs everything in
revise_beliefs
    findall((TruthValue, TKiller, Witness),

```

```

(member((TruthValue, _, _, TKiller, _, Witness), FinalTestimonies)),KL),
revise_killer(KL, FinalKiller),
motive(Motive, FinalKiller),
FinalBeliefs = [FinalKiller, Weapon, Motive],
write("My final beliefs are as follows:"), nl,
output(FinalBeliefs).

```

% There's an internal contradiction if...

```

internal_contradiction((Saw, Heard, Weapon, Killer, Motive, Witness), Testimonies) :-
    ( \+ weapon(Weapon, Saw, Heard), !); % the weapon doesn't line up...
    ( \+ motive(Motive, Killer), !); % or the motive doesn't line up...
    ( member(('_', _, _, _, _, Witness), Testimonies), !). % or they've already given
testimony.

```

```

get_truth_value(('_', _, Weapon, Killer, Motive, _), Beliefs, TruthValue) :-
    intersection([Killer, Weapon, Motive], Beliefs, Intersection),
    union([Killer, Weapon, Motive], Beliefs, Union),
    length(Intersection, ILength),
    length(Union, ULength),
    TruthValue is (ILength/ULength)+0.1.

```

```

recalculate_truth_values([], _, NewTestimonies) :- NewTestimonies = [].
recalculate_truth_values([H|T], Beliefs, NewTestimonies) :-
    H = (OldTruthValue, Saw, Heard, Weapon, Killer, Motive, Witness),
    OldTruthValue \= 0,
    get_truth_value((Saw, Heard, Weapon, Killer, Motive, Witness), Beliefs, TruthValue),
    recalculate_truth_values(T, Beliefs, RestOfList),
    NewTestimonies = [(TruthValue, Saw, Heard, Weapon, Killer, Motive,
Witness)]RestOfList].
recalculate_truth_values([H|T], Beliefs, NewTestimonies) :-
    H = (0, _, _, _, _, _),
    recalculate_truth_values(T, Beliefs, RestOfList),
    NewTestimonies = [H|RestOfList].

```

```

revise_beliefs([], [('_', _, _, Weapon, Killer, Motive, _)], [Killer, Weapon, Motive]).

```

% revise the program's beliefs

```

revise_beliefs(('_', _, _), Testimonies, [NewKiller, NewWeapon, NewMotive]) :-
    %get_killers_and_truth(KL, Testimonies),
    findall((TruthValue, Killer, Witness),
    (member((TruthValue, _, _, Killer, _, Witness), Testimonies)),KL),
    findall((TruthValue, Weapon),
    (member((TruthValue, _, _, Weapon, _, _, _), Testimonies), TruthValue \= 0), WL),

```

```

findall((TruthValue, Motive),
(member((TruthValue, _, _, _, Motive, _), Testimonies), TruthValue \= 0), ML),
sort(KL, KillerList),
sort(WL, WeaponList),
sort(ML, MotiveList),
revise_killer(KillerList, NewKiller),
revise_weapon(WeaponList, NewWeapon),
revise_motive(MotiveList, NewMotive),
motive(NewMotive, NewKiller).

```

revise_killer(List, NewKiller) :-

```

    Killers = [(L1, john), (L2, craig), (L3, theresa), (L4, matthew), (L5, daniel), (L6, victim)],
    findall((Value1, john, Witness),      member((Value1, john, Witness), List),
JAccuse),
    findall((Value2, craig, Witness2),    member((Value2, craig, Witness2), List),
CAccuse),
    findall((Value3, theresa, Witness3),  member((Value3, theresa, Witness3), List),
TAccuse),
    findall((Value4, matthew, Witness4),  member((Value4, matthew, Witness4), List),
MAccuse),
    findall((Value5, daniel, Witness5),   member((Value5, daniel, Witness5), List),
DAccuse),
    findall((Value6, victim, Witness6),   member((Value6, victim, Witness6), List),
VAccuse),
    sum(JAccuse, 0, S1),
    sum(CAccuse, 0, S2),
    sum(TAccuse, 0, S3),
    sum(MAccuse, 0, S4),
    sum(DAccuse, 0, S5),
    sum(VAccuse, 0, L6),
    modify_sum(S1, S2, S3, S4, S5, L1, L2, L3, L4, L5, List),
    sort(Killers, Sorted),
    reverse(Sorted, KL),
    get_killer(KL, NewKiller).

```

get_killer([(_ , NewKiller)|_], NewKiller).

get_killer([_ | R], NewKiller) :- get_killer(R, NewKiller).

get_killer([], _) :- false.

revise_motive(List, NewMotive) :-

```

    Motives = [(S1, adultery), (S2, money), (S3, pyromania), (S4, protection), (S5, suicide),
(S6, insanity)],
    findall((Value1, adultery),      member((Value1, adultery), List), JAccuse),

```

```

findall((Value2, money),      member((Value2, money), List),      CAccuse),
findall((Value3, pyromania),  member((Value3, pyromania), List),  TAccuse),
findall((Value4, protection), member((Value4, protection), List), MAccuse),
findall((Value5, suicide),    member((Value5, suicide), List),    DAccuse),
findall((Value6, insanity),   member((Value6, insanity), List),   VAccuse),
sum(JAccuse, 0, S1),
sum(CAccuse, 0, S2),
sum(TAccuse, 0, S3),
sum(MAccuse, 0, S4),
sum(DAccuse, 0, S5),
sum(VAccuse, 0, S6),
sort(Motives, Sorted),
reverse(Sorted, [_, NewMotive]|_]).

```

revise_weapon(List, NewWeapon) :-

```

Weapons = [(S1, gun), (S2, knife), (S3, rope), (S4, poison), (S5, matches), (S6, none)],
findall((Value1, gun),      member((Value1, gun), List),      JAccuse),
findall((Value2, knife),    member((Value2, knife), List),    CAccuse),
findall((Value3, rope),     member((Value3, rope), List),     TAccuse),
findall((Value4, poison),   member((Value4, poison), List),   MAccuse),
findall((Value5, matches),  member((Value5, matches), List),  DAccuse),
findall((Value6, none),     member((Value6, none), List),     VAccuse),
sum(JAccuse, 0, S1),
sum(CAccuse, 0, S2),
sum(TAccuse, 0, S3),
sum(MAccuse, 0, S4),
sum(DAccuse, 0, S5),
sum(VAccuse, 0, S6),
sort(Weapons, Sorted),
reverse(Sorted, [_, NewWeapon]|_]).

```

%basecase

```
sum([], Sum, Sum).
```

%sum for a 3-tuple

```
sum([(0,_,_)|Rest], CurSum, FinalSum) :-
    sum(Rest, CurSum, FinalSum).
```

```
sum([(Val,_,_)|Rest], CurSum, FinalSum) :-
    NewSum is CurSum + Val+1,
    sum(Rest, NewSum, FinalSum).
```

%sum for a 2-tuple

```
sum([(0,_)|Rest], CurSum, FinalSum) :-
    sum(Rest, CurSum, FinalSum).
```

```
sum([(Val,_)|Rest], CurSum, FinalSum) :-
    NewSum is CurSum + Val+1,
```

```
sum(Rest, NewSum, FinalSum).
```

% modify the sum of the each suspect's odds by 1.1 - their truth value

modify_sum(SJ, SC, ST, SM, SD, LJ, LC, LT, LM, LD, List) :-

```
( ( member((JTV, _, john), List), LJ is SJ + JTV); LJ = SJ),  
( ( member((CTV, _, craig), List), LC is SC + CTV); LC = SC),  
( ( member((TTV, _, theresa), List), LT is ST + TTV); LT = ST),  
( ( member((MTV, _, matthew), List), LM is SM + MTV); LM = SM),  
( ( member((DTV, _, daniel), List), LD is SD + DTV); LD = SD).
```

Appendix B: Scenarios

Scenario 1.

(blue face, nothing, poison, craig, adultery, john)
(convulsing, nothing, poison, john, protection, craig)
(nothing, banging, rope, john, protection, matthew)
(flames, screaming, matches, matthew, pyromania, daniel)
(nothing, nothing, poison, daniel, money, theresa)

Bailiff

(poison, protection)

Expected killer: john

Scenario 2:

(flames, nothing, matches, victim, pyromania, john)
(flames, screaming, matches, craig, pyromania, theresa)
(flames, nothing, matches, craig, pyromania, daniel)
(nothing, gunshot, poison, theresa, adultery, craig)
(flames, coughing, matches, victim, suicide, matthew)

Bailiff

(matches, suicide)

Expected killer: victim